Game Engines as Embedded Systems

Robert F. Nideffer, 2003

Abstract: This essay has the following mission: 1) to analyze game engines as cultural objects reflecting deeply held assumptions about what functional requirements are needed in order to experience a game in a fun and meaningful way; 2) to position the player as a key functional requirement of the engine; 3) to demonstrate how the engine and the player work together as indexical pointers to a significantly extended notion of database; 4) to question the degree to which game engines constrain or enable creative experimentation; and 5) to assert the importance of designing easy to use tools that facilitate flexible abstraction and structural modification of game worlds and game engines.

Starting Points

The attention, time and resources expended in relation to computer games and gaming emerge out of long-standing and diverse cultural traditions rooted in fundamental human needs having to do with the importance of play, interactivity and creative experimentation in our social lives. In 2002, roughly 60% of Americans over six years of age – about 145 million people – played computer and video games. Over 221 million computer and video games were sold, or almost two games for every American household. In the late 1990s, over one in four American youngsters reported playing games between seven and 30 hours per week. According to the Interactive Digital Software Association's 2002 sales information, over $6.9 billion in U.S. entertainment software sales went to games. In the past decade computer games and gaming have exploded from a niche market predominated by a particular youth demographic, to a much more diversified audience.

Games have been at the forefront of major hardware and software advances in institutions as diverse as education, entertainment, government and the military. The first CRT display was an old oscilloscope converted to display Spacewar, one of the earliest computer game examples designed and implemented by a group of graduate students working at MIT in the early 60s, in a lab funded by the military for the purpose of calculating missile trajectories.

IMAGE 01

Spacewar: 1961

Spacewar also catalyzed the development of the first joystick as a controller, modeled after a control device used by the MIT Tech Model Railroad Club. Legend has it that the UNIX operating system was developed by Ken Thompson on a PDP-1, largely out of a desire to play the game in a locally networked environment (Graetz, 1981)[1].

IMAGE 02

Spacewar Hardware: 1961

The connection of games and gaming to the military runs deep. It's common knowledge that Operation Desert Storm was prepared for by doing simulation strategy exercises in Florida prior to the invasion of Iraq, and that the US military is currently pumping large amounts of capital into figuring out how to appropriate gaming principles for battle training in massively multiuser SimNet environments. Such synchronicities between games, military preparedness, and academically driven R&D recently achieved new heights (or plunged to new depths, depending upon your point of view) in 1999 when the U.S. Army awarded a five-year contract to the University of Southern California to create the Institute for Creative Technologies (ICT).

As described on their public website, "the ICT's mandate is to enlist the resources and talents of the entertainment and game development industries and to work collaboratively with computer scientists to advance the state of immersive training simulation" (ICT, 2003)[2]. Essentially the military want to implement training environments that have a narrative and emotional impact on par with the best that Hollywood has to offer. One thing that this alliance clearly indicates is the degree to which technology R&D within the private sector, and specifically the commercial games industry, has outpaced what goes on in the government sponsored labs of academia and the military. The ICT, by its mere existence, is an explicit acknowledgement of this transition.

It doesn't take a rocket scientist to see that electronic gaming is transforming the entertainment marketplace, and increasingly dominating the time and attention of children and adults. What analog film and television represented at the close of the 20th century, computer games and interactive entertainment represent at the dawn of the current century. While all manner of entertainment will continue to exist in a myriad of forms and fashions, it's not too difficult to see that we are shifting from relatively passive media experiences to more interactive ones, from being simple consumers of digital media product to playing

active roles in the creation of that product, and from computing in small scale individually isolated domains to computing in large scale collectively networked ones.

As we undergo these shifts, we must begin critically situating computer games in relation to the cultural milieu in which they get produced, distributed and consumed – and give them the same kind of attention and analysis we give to art, literature and film. This process entails developing methodologies and theoretical vocabularies that recognizes the uniqueness of the medium, while at the same time places it in the context of the social conditions that produce it. In this light, one of the most important things to consider is how technical infrastructure can influence or even dictate form and content – for games, as well as for other creative work that incorporates game design principles, metaphors and technology.

Engines of Creation

So what exactly is a game engine and how does it work? Simply put, a game engine is a piece of software that provides the technical infrastructure for games. As pointed out in an informative essay published by Steven Kent in *GameSpy*, the engine is responsible for rendering everything you see and interact with in the game world. Game engines not only provide the rendering, they can also provide the physics models, collision detection, networking, and much of the basic functionality the player experiences during game play. The game engine should not however be confused with the game itself. This would, as game developer Jake Simpson points out, be analogous to confusing the car engine with the entire car. A key difference is that "you can take the engine out of the car, and build another shell around it, and use it again" (Simpson, 2002)[3]. The same holds true, more or less, with game engines and game applications. However, the customization needs of particular titles, coupled with the proprietary interests of publishers, often dictates that developers code their own engines in-house, or significantly modify licensed ones to suit their specific interests. In either case, considerable amounts of hard-core sweat, tears, and programming are usually involved.

In the last several years game engines have advanced by leaps and bounds in terms of things like memory management, processing power, optimization procedures, networking capabilities, and pushing polygons to screens. If you think back to the early days of 3D computer games, characters like those found in Castle Wolfenstein (circa 1992) had polygon counts of several hundred (if that), whereas today it's not uncommon for characters to have polygon counts in the in the thousands, as is true for DOOM III, with the source models (often used for promotional materials, and transitional animations during play) to have polygon counts in the millions (Kent, 2002)[4].

IMAGE 03
Castle Wolfenstein: 1992

IMAGE 04
DOOM III: 2003

When you're talking about 3D game engines, until very recently polygons remained developers' primary concern. How many polygons an engine allows you to efficiently use largely determines the perceived realism of the environments you're creating. However, one of the interesting things happening as engines and hardware evolve is that: "most designers agree that raw polygon counts – so important in generations past – are no longer the main measure of a graphics engine. Instead, more and more computing power is needed to realistically light, shade, and texture those polygons to make objects appear even more real" (Kent, 2002)[5]. Legendary engine programmer John Carmack, who played a lead role in ushering in the game genre known as the "first person shooter," and is the programming mastermind behind Castle Wolfenstein, Doom and Quake, and the ID Software company, reinforced this notion when he stated that what he wanted wasn't more polygons, but *more passes* per polygon, since with each "rendering 'pass' a computer makes before it displays the graphics, more and more detail can be added" (Kent, 2002)[6].

Rendering and the Holy Grail of Realism

As Kent and others have pointed out, improved lighting, increasingly accurate physics models, and more believable, artificial intelligence (AI), are seen as the next frontier for game engines by many in the industry. For anyone who attends things like the Game Developers Conference (GDC), the Electronic Entertainment Expo (E3), or SIGGRAPH, it very quickly becomes apparent that these concerns are voiced almost exclusively from the desire to enhance the game world's realism. Game AI is particularly interesting to think about in these terms. In the game world AI tends to be associated with what are called NPCs, or Non-Player Characters. NPCs are pre-scripted machine controlled characters that the player-controlled character interacts with. Good NPC AI will allow the NPC to function, or even more importantly *learn* how to function, in context specific ways, adding impressive texture to the gameplay. The behavioral models applied to the complex coding of this kind of context-specific conduct, not surprisingly, come from careful observational studies of human interaction and human learning. Thus even

things like inappropriate or random behaviors or mistaken responses may be hard-coded into the system to more believably mimic human situations.

There is a general consensus that in order for the game development community to expand beyond the relative niche (though incredibly lucrative) market is currently occupies, it will need to do things like diversify genre and content, and make more compelling characters and narratives. As game engines and hardware platforms become capable of in-game photorealism or even what's called enhanced realism (that which is beyond photo-realistic), it is assumed that players will more readily be immersed and identify with character and story – as tends to be the case with good literature, films and television – making for a deeper and more meaningful media experience.

In many ways this cultural moment within the game development and graphics community can be likened to what was happening in the arts, particularly within the domain of painting, prior to the advent of photography. Portrait and landscape painting dominated. There were huge commissions, and the more detailed and realistic the work, the more value it had, and the more compelling, beautiful, and aesthetically pleasing it was deemed to be.

IMAGE 05
Rembrandt van Rijn: Nicolaes Ruts

Then along came the camera. Artists had been using camera obscuras of various kinds throughout the 16th, 17th, and 18th centuries, in order to form images on walls in darkened rooms for them to trace. But it was a combination of Adolphe Disderi's development of *carte-de-visite* photography in Paris, leading to worldwide boom in portrait studios for the next decade (1854), the beginning of the stereoscopic era in the mid-1800s, and the subsequent popularization of photographic technology developed in the late-1800s and early to mid-1900s thanks largely to George Eastman and Kodak, that really heralded a new era. The camera was the most faithful documentarian imaginable. No longer was it necessary to so faithfully attempt to mimic an external reality with paint. From a conceptual and creative standpoint, needless to say, things started to get more interesting.

IMAGE 06
Pablo Picasso: Daniel-Henry Kahnweiler

Artists began to more boldly experiment with shape, color, light, texture and form. As a result of this process, some of the most influential art historical movements of the 20th century happened – impressionism, post-impressionism and pointillism (1870s-1880s), fauvism (1905-1908), die brucke (1905-1913), der blaue reiter (1911-1914), expressionism (1905-1925), cubism (1907-on), dadaism (1916-1922), surrealism (1924-1930), de stijl (1917-1931), abstract expressionism (1940-1950), color-field (1945-1950), pop art (1950s), minimalism (1958-on), op art (1965-on), and so on. The list could continue. The point here is that with the introduction and adoption of a specific technology (i.e., the camera) the aesthetic sensibilities of the dominant art-world culture shifted and diversified, and whole new horizons of possibility opened up to experimentation and creative exploration. The same type of conceptual shift needs to happen in the context of computer games and game engines (Greenspun, 2003)[7].

Database Interfaces

You might think of the game engine as a database interface – a mechanism through which a pre-determined, relatively constrained collection of procedures and protocols are used to render a world and make it navigable in context. If we wish to look at the game engine as a cultural artifact that circulates within a specific social domain, then we must extend the boundaries of what strictly constitutes the game engine and posit the game player as not only a functional requirement of the engine, but as its key constitutive element. Without the player, the engine effectively ceases to exist. Once the player is positioned as an integral part of the game engine, a pivotal question becomes what then constitutes the database utilized in the game engine's rendering of the game world?

The classical computer science definition of database refers to any organized store of data for computer processing. Data is taken to be items upon which operations may be performed within a computing environment. It's not too much of a stretch then to claim that any of a game's resources – the images, models, textures, sounds, interfaces, codebase – can be thought of as elements of the database, as they are all part of an organized store of data upon which the game engine performs operations, primarily though not exclusively in relation to the user input. The methods used to extract resources from the database play a big role in the generation of contextually specific meaning, a generative process that is required in order for goal-oriented interaction to occur. The more resources there are to pull from, and the more robust the methods for pulling, the greater the amount of context that can be provided, and the

richer the potential exchange. Without such additional context we run the risk of less functional reading, navigation and orientation to the mental and physical space we occupy.

It is equally important to reflect upon the process of populating (asset creation), accessing (asset management), rendering (asset visualization), manipulating (asset modification) and playing a given game, if we want to try and tell a tale about the underlying logic. However, that underlying logic, particularly with regard to complex technical infrastructures can only usually be uncovered to a limited degree. Knowledge, and the narrative or meaning making mapped upon experience in relation to that presumed knowledge, is always, situated, contextual, and partial. This partial awareness is compounded by the fact that the dominant paradigm in software engineering is to keep the nuts and bolts of infrastructure hidden from the end user.

For example, imagine Kanji, the character set of the Chinese and Japanese language, as a database. Within Kanji there are over 50,000 unique characters. Roughly 2,000 or so are used in daily conversation. Let's assume you've been studying the language and know about 300 of those characters. You could probably get by reasonably well if you visited China or Japan, and do things like read bits of the newspaper, order in restaurants, find your way to the bathroom, and so on. However, your relationship to the culture would be fairly superficial, as your contextual understanding – the ability to make connections between ideas, appreciate subtle nuances, wordplays, humor, sophisticated concepts, analogies and the like – would be rather severely limited. In other words, you wouldn't be a very robust interface to the database.

But we also need to ask, and this is absolutely critical, what constitutes the database *of the player*, where does it begin and end, and how does it get rendered during play? To begin answering this question, a concept proposed by Berkeley sociologist Ann Swidler, who is responsible for advancing the notion of "culture as a toolkit" could prove useful. From Swidler's perspective, individuals draw on cultural tools to solve problems and interpret their social worlds. There are any number of different cultural values and beliefs in an individual's cultural toolkit depending upon the various environments and experiences in which the individual is situated (Swidler, 1986)[8].

For our purposes here, think of the player's cultural toolkit as a kind of database; and in addition to values and beliefs, include things like personal history, geographic location, language, education, race, class, and ethnicity, as constitutive elements of that database, all of which influence the way meaning making happens during not only the gaming experience, but also during the development of the infrastructure designed and implemented to support that experience.

Every game has its set of rules, grammar, and logic. Together these constitute the language or narrative of the game, a language or narrative that must be learned in order to gain experience, increase levels, acquire power, or indeed make meaning within the given system. Interpretation of the world must happen continually in order to negotiate action within it. Thus players as well as developers inescapably function as indexical pointers to their respective cultural toolkits or databases at all phases of game engine and game application production, distribution and consumption.

Give A Boy A Hammer…

The metaphor of the cultural toolkit provides a useful mechanism for thinking about how we may extend the idea of the database into the social domain. It also allows us to examine how the relevant attributes and elements of the database can mutate in relation to context. Yet for all its utility, the toolkit analogy is not as useful in explaining why certain gaming designs, aesthetics and experiences are consistently chosen or privileged over others, or why certain constraints are continually coded into software. Frame theory provides a more useful theoretical scaffold for addressing these concerns. A core concept of Frame theory is "frames of reference," defined in the *Dictionary of Modern Thought* as "the context, viewpoint, or set of presuppositions or of evaluative criteria within which a person's perception and thinking seem always to occur, and which constrains selectively the course and outcome of these activities" (Bullock, etc, 1988)[9]. A wide range of theoreticians within the social sciences have, in varying degrees, mobilized this concept to explain human behavior within the disciplinary domains of psychology (Gregory, 1972; Hochberg, 1987; Piaget, 1973; Zimbardo and Leippe, 1991)[10], sociology (Benford and Snow, 2000; Gamson, Croteau, Hoynes, and Sasson, 1992; Goffman, 1975; Schutz, 1962-1966)[11], and anthropology (Bateson, 1954, 1973; Whorf, 1956)[12] (See Atherton, 2002 for more detailed discussion)[13].

Admittedly, there is some slippage in the usage of the terms *framing* and *ideology*. At the risk of opening Pandora's box, let's postulate ideology as a complex web of systematically and institutionally related ideas, values and norms that is often seen as having a material basis as it articulates the social world and positions subjects within it. Ideological systems have tended to be theorized as inflexible and resistant to change. On the other hand, framing more readily allows for a give and take to be injected into rigid and unidirectional understandings of ideology, so characteristic of the earlier literature on the topic. As such,

frames can become tactically and strategically mobilized when consciously utilized in effort to realign prior framings that have become relatively fixed and stabilized (Johnston and Oliver, 2000)[14]. Good frames (the objects), and effective framings (the objects put into action), have a cultural resonance, meaning they are in synchrony with a collective or shared belief about how the world works, albeit a belief that is often hidden and taken for granted until exposed and/or threatened with a competing frame.

Arguably, in order for gameplay to be successful and resonate with the player in the ways anticipated and intended by the developers, the proper cultural toolkits and frames of reference need to be available to the player in order for appropriate character identification, narrative and environmental navigation, and contextual meaning making to occur. However it should by no means be assumed that there is a one-to-one correlation between the cultural toolkits and frames of reference drawn on by the producers of the game, and those employed by the consumers of that game in order for a game to be perceived a success in terms of design or economics, or at a minimal level, simply as fun. It does however imply that if more tools are shared, and the frames of reference deployed within the game space are meaningful to the player, the gameplay experience will have a far more profound emotional, psychological, and physiological impact. This implication holds particularly true as game concepts, technology, and industries mature and diversify.

Similarly, in order for the success of the game engine to be maximized with the developer community, it also must resonate with the popular frames of reference consistently mobilized within that domain of interest. In other words, when developers, and by extension players, have reasonably consistent values, beliefs, and expectations with regard to what qualities it's important for engines and the applications wrapped around them to possess, a game, and the engine supporting it, will have popular appeal. And when those values, beliefs, and expectations are not shared, the appeal will be far less. This dynamic can easily be seen in the many failed efforts to market games cross-culturally, particularly when there is little sensitivity to the specificities of a region's people and history. Though it's beyond the scope of this essay, the converse situation is perhaps even more interesting: how do we explain the cross-cultural success of particular game titles that have little to do with or even run counter to the particularities of the people and places in which they get played, and what significance do we make of it? This is of course a variation on a theme that has been relevant to the study of media hegemony from its inception.

Granted, the purpose of this essay is not to explicate the various toolkits, frames and framings that have been present within the computer game development community over the past several decades, particularly as they percolate up and manifest at the level of the game application. That task gets taken up elsewhere (Nideffer, 2003)[15]. The goal here is simply to present the concept in effort to explain how and why certain functionality gets consistently coded into software, and the difficulty of creatively modifying that functionality once it's been embedded into infrastructure. It's also to begin providing a theoretical foundation that can be useful for deliberately thinking through alternative frames and framings in effort to enhance creative production and consumption possibilities, diversify the rapidly expanding media ecology tied to games and gaming, and effect responsible social change.

And the World Becomes a Nail

What are some of the key functionalities that are perceived necessary within the development community in order to provide the game designer with the requisite tools for efficient and successful product as it hits the competitive marketplace? To address this question, we'll restrict the discussion to 3D game engines, though much of what follows could be as readily applied to 2D engines, and begin by looking at it in the context of the Electronic Entertainment Expo (E3), held annually in Los Angeles, California to showcase the latest in games technology.

One of the most noticeable things when walking around the exhibition floor of E3 is the remarkable functional and aesthetic resemblance between titles utilizing 3D engines. There's almost always an animate thing rendered to a display device that gets situated in a largely recognizable wire-framed and texture-mapped environment; its movement is controlled through a hardware interface such as a joystick, keyboard, or game pad; it has a predictable real-world physics model attached to it; it's proximately aware of other things around it or that it comes into contact with; there will be other things similar to it that are either remotely controlled by other players within a networked social space or are pre-coded entities; it has ambient and event based sounds, lighting and particle effects tied to its location and action; it usually gains power and privilege in the system over time with successful manipulation of objects and completion of goals and objectives; and temporally and spatially sensitive state information can be saved on it so that it's possible to stop and start again from where last left off. Clearly, this list could continue at great length. Why are these attributes seemingly so entrenched? Arguably, it has less to do with technical limitations than it does with conceptual ones.

Granted, we can push more pixels to the screen, have higher resolution static and dynamic imagery, navigate more complex environments, and move around in 3D. And while we can see considerable

variation with the game shell – the game content wrapped around the engine – there still has been surprisingly little experimentation at the level of the functional attributes assumed to be integral to the successful 3D game engine. And without that flexibility at the level of the infrastructure it's difficult if not impossible to see radical transformation happen at the level of the application hooked into that infrastructure. These things look and feel strikingly similar.

IMAGES 07-09
First Person Shooters

Unreal Tournament	Quake III	Half-Life

Even in the realm of the MMORPG (Massively Multiuser Online Role Playing Games), most of what you find can be linked directly back to text-based MUDs (Multi-User Dungeons or Domains) and MOOs (Multiuser Object Oriented domains) which witnessed impressive popularity in the 1980s and early 1990s. MMORPGs like Everquest, Ultima Online, Asheron's Call, and the Sims can functionally and conceptually be thought of as extended versions of MUDs and MOOs that have been given a graphical front-end.

IMAGES 10-12
MMORPG

Everquest II	Ultima Online	Asheron's Call II

There are a lot of reasons for this lack of structural experimentation. Some of the reasons have to do with *generic convention* – and how once something establishes itself as successful in the marketplace, inertia sets in. Bigger budgets and longer production times are required to remain competitive as the industry expands. This of course leads to an aversion to risk-taking, particularly in less established production houses. Blockbuster Hollywood filmmaking is an excellent example of what happens when the economics of production start to put a stranglehold on creative experimentation. The games industry is rapidly following suit, to the consternation of many developers, publishers and players.

Other reasons have to do with the relative openness and ease of use of the software and hardware required to implement a working game, let alone modify or creatively mess around within the confines of an existing one (discussed in more detail below), either at the level of the application or the underlying engine driving it. And of course, perhaps paramount are the constraints that result from the utilization of specific cultural toolkits, frames and framings.

For illustrative example, imagine a "Repetition in Game Engines and Game Design" continuum, where on one end you plot "Materialist Reasons" and on the other "Ideational Reasons" for why commercial (as well as independently produced) product looks and feels so similar. Game industry economics and the openness of software and hardware would be positioned way down toward the materialist end of the continuum, while cultural toolkits, frames, and framings would be positioned toward the ideational end of things. Predictably, as is the case currently, when the material and the ideational are working in consort, change is quite difficult to come by, and one cannot expect to find much diversity or heterogeneity.

For instance, take the *idea* that modeling game physics after real-world physics is necessary in order for gameplay to be fun, or that characters need to be recognizable and anthropomorphic for identification to happen more intensively, or that photo-realism allows for increased immersion in the game world. Couple that with an awareness that all of these functional attributes are present in the most economically profitable titles and it makes it exceedingly difficult to mess with the prevailing codes and conventions. This holds true for both commercial developers as well as independent producers. However, there is some small but significant indication that the tide may be turning.

Cracks in the Edifice

Something interesting has started to happen in the gaming industry recently. Tools that had been used exclusively by title developers are now being released to the general public in order to allow players to customize and/or radically modify their game environments. Such creatively messing around within the confines of existing games – a.k.a. "modding" – has become a fairly widespread phenomenon within the videogame community. Game modding tends to consist of players who possess a facility for programming, and who create custom level maps, character skins, weapon types and retool various other objects and items that are part of the game.

As veteran games programmer and author Jake Simpson points out, game mods came about from the editing programs that enabled gamers to modify the original .WAD files for Doom – basically files that contain all the information about the graphics, sound, level maps, etc. for the game – and supply their own home-brewed level designs and textures. Gamers started playing with these custom-built tools and

found they too could produce levels that other people wanted to play. It wasn't long before game companies, notably ID software, noticed this trend and took it a stage further with the Quake series of engines, designing the game so that it was eminently user modifiable. ID even went so far as to release their own design tools, instructions, and code samples, so aspiring game programmers could tweak the Quake Universe (Simpson, 2002)[16].

Other companies soon followed suit, and started building modification tools into their own game titles in order to see what players would do, and to assess where future development efforts might be focused. Games like Doom, Quake, Unreal Tournament and Half-Life are all now able to bring out users' creativity by providing level editing, mod authoring, and server tools to players; the hope is that eventually the whole thing will just take off on its own momentum (Stonewall, 2000)[17].

One of the most interesting examples of just how far the game mod can go is the game Counter-Strike. Counter-Strike is a modification to another Counter-Terrorism game called Half-Life (mentioned briefly above). It modifies the multiplayer aspects of Half-Life to bring to it a more team-oriented gameplay. As of March 9, 2003, Counter-Strike had the largest service footprint of any online action game with roughly 35,000 servers generating over 4.5 billion player minutes per month (Official Website, 2003)[18]. What is even more impressive than the sheer volume of traffic these numbers indicate is that Counter-Strike became far more successful than Half-Life, the game it was originally modified from – in fact so successful that Valve, the parent company of Half-Life shrewdly decided to acquire Counter-Strike and continue developing it in-house.

Another development worth watching is the introduction of a whole new genre of real-time movie making called "Machinima" which uses 3D multiplayer game engines as the primary authoring platform. Similar to the modding phenomenon described above, developers are now designing custom tools to support this highly creative and entirely unintended use of the game space. A blend between film, animation and gaming, filmmakers with a home PC work exclusively within a digital realm to create feature-length movies that just a short time ago would have required millions of dollars to make using traditional computer graphic techniques. Once the film is made, Machinima productions can be distributed over the Internet and rendered on any viewer's computer (Machinima.com, 2001)[19].

Many of today's game gurus came from this early modification experience, and whole new industries have been spawned in response to the modding community. Additionally, modding competitions have begun cropping up, art exhibitions themed around game hacking and modding have been done, and "how-to" courses, panels and workshops on modding are being conducted at universities and conferences around the country. The proponents of this "revolutionary" new medium are so full of hope and optimism that an Academy of Machinima Arts & Sciences has formed (currently headquartered in New York) and will play host to the 2003 Machinima Film Festival (Academy, 2002-2003)[20].

But let's be real for a moment. The film community has little to fear, at this point. Even the "best of" Machinima films are so severely constrained by the technical infrastructure, and struggle so desperately against what's been coded into the system, that it often becomes painful to watch. So far, making compelling movies with a game engine is like trying to perform Swan Lake with a SWAT team. No matter how much effort gets put into the production, if you're expecting classical ballet, you won't be very satisfied.

Granted, Machinima may be interesting on other levels, such as the methods and modes of production and distribution, or the intentional misuse of technologies intended for other purposes, and the like. But the current fascination has more to do with the *idea* of using a game engine for movie making than the *reality* of what gets produced. The only logical way to understand the hype is to assume that it comes from folks who have never sat and tried to download and watch any.

That said, it's impressive to see what people are capable of doing with tools that until very recently were never meant to support the use to which they're being put. While modding and Machinima may provide a glimmer of hope for those who aspire to creatively play with game designs and technologies in a deeper way, the larger issue remains one of how open-ended and far reaching the tools facilitating that experimentation actually are.

Winding Up and Down and Back Around

In the history of the arts, the most inspiring, uncomfortable and transformational moments have tended to be those that radically broke with tradition and were subsequently associated as part of an avant-garde. They were moments that catalyzed more than simple variations on pre-established practice and content. Instead they were somehow able to dig deeper and to fundamentally alter form. This also consistently meant that they would be perceived as threatening to those who occupied seats of

institutional power and prestige. The result was that the institutions and cultural gatekeepers either shifted to accommodate and appropriate the avant-garde or they collapsed and were replaced.

Developing a facility for taking a critical distance in relation to work and play, to a certain degree requires keeping one's own cultural practices and frames of reference "anthropologically strange" – meaning to consciously reflect to whatever degree possible upon one's embedded behaviors, values, and beliefs as "foreign" or "other." When successful this process helps to enable a type of creative production that disrupts social space in the interest of rendering visible the taken-for-granted and invisible structures of language, meaning, and power. Of course in the commercial world such critical consciousness is not necessarily so openly embraced – at least not until it's seen as useful in marketing to an under-exploited consumer base. Regardless, even when the goals are not so overtly political, cultivating this awareness allows for far more responsible, subtle and sophisticated output. This holds especially true when thinking about infrastructure design, where the primary desire has been to actively code to support things such as "transparency" and "intuitive interface" at every level during the game's development process.

It takes a lot of effort to cast light on the internalized dynamics motivating and enabling meaningful action in the world. But once those dynamics start to be revealed and situated in a way that speaks persuasively to the heart and the mind, change becomes possible at both an individual level as well as an institutional one. We need to develop tools and techniques that facilitate that process of uncovering in the interest of strategically assessing the past, consciously being in the present, and responsibly diversifying the array of future possibilities. From the beginning of human history play and games have been a key way in which we learn about the rules and regulations that govern our social lives.

Hopefully it has become clear that the game engine is not simply software, it's software that reflects and embodies the cultural conditions symptomatic of the developers of the system, as well as the end users of that system. Moreover, when players are positioned as integral components of the system, the question of database, what it is or where it begins and ends, gets radically transformed. The database, and the aesthetic manifest through the game application in relation to the database as it gets rendered by the engine, consists not only of the more traditionally conceived content of the game – the images, models, textures, sounds, interfaces, codebase – but of the cultural toolkits and frames of reference brought to bear on the design, implementation and play of the game as well.

The greater the degree to which one's cultural conditions are brought into critical consciousness, and the more abstracted, accessible and flexible the interplay between the software, hardware, and player is made, the richer the creative power and possibilities will be when that infrastructure gets put to use. The problem is that reflexivity in relation to the ways that ideological systems get coded into infrastructure is practically nonexistent. Clearly, we have a lot of work to do.

[1] J. M. Graetz, 1981. "The Origin of Spacewar." Creative Computing Magazine, 1981, reprinted online at http://www.wheels.org/spacewar/creative/SpacewarOrigin.html
[2] "Welcome to the ICT," printed online at http://www.ict.usc.edu/disp.php?PHPSESSID=869044eb898e4f396222ffde8c5c179f, 2003.
[3] Jake Simpson, "Game Engine Anatomy 101, Part I : Intro to Game Engines, The Renderer, and Creating a 3D World," printed online at http://www.extremetech.com/article2/0,3973,594,00.asp, April 12, 2002.
[4] Steven L. Kent , "Engines And Engineering: What to expect in the future of PC games," GameSpy.com, printed online at http://www.gamespy.com/futureofgaming/engines/, Oct. 31, 2002.
[5] Steven L. Kent , "Engines And Engineering: What to expect in the future of PC games," GameSpy.com, printed online at http://www.gamespy.com/futureofgaming/engines/, Oct. 31, 2002.
[6] Steven L. Kent , "Engines And Engineering: What to expect in the future of PC games," GameSpy.com, printed online at http://www.gamespy.com/futureofgaming/engines/, Oct. 31, 2002.
[7] Philip Greenspun. "History of Photography Timeline," printed online at http://www.photo.net/history/timeline, 2003.
[8] Ann Swidler, "Culture in Action: Symbols and Strategies." American Sociological Review, 51 (April): 273-286, 1986.
[9] Alan Bullock, Oliver Stallybrass, Stephen Trombley, Bruce Eadie (Eds), 2nd edition, The Fontana Dictionary of Modern Thought, (Fontana Press, 1988).
[10] Julian Hochberg, "Gestalt Theory" in R L Gregory (ed). Oxford Companion to the Mind, (Oxford: Oxford University Press, 1987); Jean Piaget, The Child's Conception of the World, (London: Paladin, 1973); Phillip G. Zimbardo and Michael R. Leippe. The Psychology of Attitude Change and Social Influence, (New York: McGraw-Hill, 1991).
[11] Robert Benford and David Snow, "Framing Processes and Social Movements: An Overview and Assessment," Annual Review of Sociology, 26: 611-639, 2000; William A. Gamson, David Croteau, William Hoynes, and Theodore Sasson, "Media Images and the Social Construction of Reality," American Review of Sociology, Washington, D.C.: American Sociological Association, 18:373-393, 1992; Erving Goffman, Frame Analysis, (Harmondsworth: Penguin, 1975); Alfred Schutz, Collected Papers, vols I – III, (The Hague: Martinus Nijhoff, 1962-66).
[12] Gregory Bateson. Steps to an Ecology of Mind, (London: Paladin, 1972); Benjamin L. Whorf, Language, Thought and Reality, (Cambridge, Mass: MIT Press, 1956).
[13] James Atherton. On Learning to "See", printed online at http://www.doceo.co.uk/tools/frame.htm, 2002.
[14] Hank Johnston and Pamela Oliver, "Mobilization Forum: A Reply to Snow and Benford," Mobilization: An International Journal, 2000, 5(1), 61-63, reprinted online at http://216.239.57.100/search?q=cache:OFMcTwefh3YJ:www.ssc.wisc.edu/~oliver/PROTESTS/ArticleCopies/reply_finalrevisionfrom%2520HJ.pdf+framing+benford+snow&hl=en&ie=UTF-8.
[15] Robert Nideffer, "Framing Videogames," Context Providers, (Cambridge, Mass: MIT Press, Forthcoming, 2003).
[16] Jake Simpson, "Game Engine Anatomy 101, Part I : Intro to Game Engines, The Renderer, and Creating a 3D World," printed online at http://www.extremetech.com/article2/0,3973,594,00.asp, April 12, 2002.
[17] Stonewall, "RPOV," Interview with Tim Sweeney, Founder and President, Epic Games, printed online at http://www.r-pov.com/interviews/tsweeney2.html, March 21, 2000.
[18] "Valve Opteron Support -- Sunday, March 9, 2003," Official Counter-Strike Website, printed online at http://www.counter-strike.net/, 2003.
[19] "Introduction to Machinima", Machinima.com, printed online at http://www.machinima.com/Whatis/intromach.shtml, 2001.
[20] Academy of Machinima Arts & Sciences, printed online at http://www.machinima.org/, 2002-2003.